SYSTEM AND METHOD FOR TESTING TIME-VARYING SIGNALS

INVENTOR

Art Gorman

CERTIFICATE OF MAILING BY "EXPRESS MAIL"
UNDER 37 C.F.R. § 1.10

"Express Mail" mailing label number: **EL 622 696 859 US**
Date of Mailing: August 28, 2001

I hereby certify that this correspondence is being deposited with the United States Postal Service, utilizing the "Express Mail Post Office to Addressee" service addressed to **Box PATENT APPLICATION, Commissioner for Patents, Washington, D.C. 20231** and mailed on the above Date of Mailing with the above "Express Mail" mailing label number.

[illegible]

Johann S. Mercado
Signature Date: August 28, 2001

SYSTEM AND METHOD FOR TESTING TIME-VARYING SIGNALS

INVENTOR:

Art Gorman

COPYRIGHT NOTICE

CLAIM OF PRIORITY

[0002] This application claims priority to Provisional patent application Serial No. 60/300,512, filed June 22, 2001, entitled SYSTEM AND METHOD FOR TESTING TIME-VARYING SIGNALS.

TECHNICAL FIELD

[0003] The present invention relates generally to the testing and generation of signals that may vary over time.

BACKGROUND

[0004] Java Management Extensions (JMX) define a community process standard for managing resources and remotely accessible agents. JMX defines an architecture for network and/or application management in the Java programming language, as well as design patterns, services, and APIs. JMX utilizes the standard industry model Managed Beans, or MBeans, for use in Java programs and management applications. MBeans are Java objects that implement application resources. These MBeans may be managed by a JMX agent or other agent supporting the MBean concept. A set of complimentary services is typically specified, which work with these MBeans to monitor and manage Java-based applications. A monitoring Mbean, or JMX monitor, may be dynamically loaded by an application, and may send an event or notification when the component or parameter being monitored reaches a certain value or state.

[0005] JMX monitors typically detect and monitor signals that vary over time, functioning similar to a digital sampling oscilloscope. When developing these monitors, it is desirable to be able to test not only for accuracy, but also for stress and load capabilities. Digital signal generators of the prior art are typically used in the analog realm, and lack the necessary JMS format capabilities needed to test these JMX monitors. These prior art signal generators also fail to generate unorthodox signals, such as a string of words.

[0006] It is therefore an object of the invention to develop a method and system for testing a JMX monitor that can run for an extended duration, allowing for controllable stress and load testing.

SUMMARY

[0007] The present invention includes a system for testing signal monitors, such as JMX monitors. The system utilizes a generator bean to generate a signal. A user may specify an equation and/or parameters to be used in generating an appropriate signal. A monitor bean is used to monitor the signal being generated. A notification is created by an MBean in response to the monitoring, such as may notify a user or application that the signal has reached a certain value, or may store the current value to a data store.

[0008] Also included in the present invention is a method for testing a JMX monitor. In the method, a signal is generated using a generator bean, such as a signal generator Java MBean. Equations and/or parameters may be specified in determining the type of signal to be generated. The generator bean is then polled at a frequency at least twice the frequency of the generated signal using a monitor MBean of the JMX monitor. A testing value is then returned for each polling of the generator bean.

BRIEF DESCRIPTION OF THE FIGURES

[0009] **Figure 1** is an illustration of an agent architecture in accordance with one embodiment of the present invention.

[0010] **Figure 2** is an example sequence diagram for a test in accordance with one embodiment of the present invention.

[0011] **Figure 3** is a diagram of a testing process in accordance with one embodiment of the present invention.

[0012] **Figure 4** is a flowchart showing the testing steps of one embodiment of the present invention.

[0013] **Figure 5** is a diagram of a testing system in accordance with one embodiment of the present invention.

DETAILED DESCRIPTION

[0014] A signal generator in accordance with the present invention comprises an MBean, hereinafter referred to as a SignalGeneratorMBean. A SignalGeneratorMBean is an interface that conforms to the static MBean design patterns. This embodiment includes classes derived from an application, such as SignalGenerator, which generates the desired signal(s). Equations can be used by the SignalGeneratorMBean to precisely define the desired testing signals to be used. Users can specify or set values and select the appropriate equation(s) for each testing situation. The equations can be entered by the user, stored in a library or file, hard coded into the application, or accessed by any other appropriate means. Users may select or set values by such means as direct entry, selection, or from a file or library containing predetermined values.

[0015] In one approach for JMX monitoring tests in accordance with the present invention, the sequence of events may be enforced by a class, such as for example MonitorMBeanTestDriver. Steps of such a sequence are shown in **Figure 4**. In this embodiment **400**, the class includes four parameters: (1) SignalGeneratorMBean for generating the signal, (2) MonitorMBean for monitoring the generated signal, (3) NotificationListener to listen to MonitorMBean and generate a notification upon certain conditions, and (4) StopTime to designate the duration of the test. These parameters, as well as others, can be specified for each test.

[0016] Initially, SignalGeneratorMBean is registered with the system and setup for testing **402**. MonitorMbean is also registered and setup appropriately **404**. MonitorMBean is attached to SignalGeneratorMBean **406** such that MonitorMBean can monitor the generated signal. NotificationListener is attached to MonitorMBean **408** so a notification can be generated for appropriate circumstances, such as MonitorMBean detecting a signal surpassing a threshold value. Once the system is setup, MonitorMBean can be started **410**. SignalGeneratorMBean is also started **412**. SignalGeneratorMBean is allowed to run for a set time, herein referred to as StopTime **414**. After the set time has passed, SignalGeneratorMBean and MonitorMBean are stopped **416**. All events are checked to see if they correctly fired to the client **418**. If so, MonitorMBean is deregistered **420**.

[0017] One view of a system **500** in accordance with the present invention is shown in **Figure 5**. In this embodiment, a set of equations **502** is used to input a desired signal form to the signal generator **504**. The signal generator **504** generates a signal according to the input equation, which is received by a monitor **506**. The monitor **506** is in communication with a notifier **508**. The monitor can send a signal to the notifier in one embodiment, while the notifier can poll the monitor in another embodiment. When the monitor sends a response or signal to the notifier regarding measurement of the signal, the notifier can send a notification to a listener **510**, which is adapted to receive the notification and act accordingly, such as by sending a message to a user or recording a data point.

[0018] Some tests may be time sensitive. In one embodiment, the failure to meet a timepoint for a particular test, such as may be due to network latency or logic server errors, can cause the test to abort. Other embodiments may utilize an error handling routine, throw an exception, or utilize any other appropriate error handling approach. Any errors can also be logged, such that the resultant logs can later, or concurrently, be analyzed.

[0019] The functional blocks for the embodiment **100** shown in **Figure 1** may be mapped to a server, such as a logic server ("LS"). The MBean server **106** on the agent side **102** of the figure corresponds to an LS. The JMX managed resource **108** on the left is an MBean that corresponds to a service component, such as an Enterprise Java Bean or Java Messaging Service. The MBeanServer **106** can be accessed by any appropriate method known in the art, such as by performing a JNDI (Java name directory interface) lookup. The MBeans, including the Monitor MBean **122** and Query MBean **124**, can be retrieved with an exact lookup or with a query, such as may be based upon an MBean ObjectName.

[0020] The tests described in this embodiment correspond to a JMX-enabled management application **110**, as may be contained in a Java virtual machine **112**. The JMX-enabled management application **110** is shown on the manager side **104** of **Figure 1**. Communication between the manager-side Java virtual machine **112** and the agent side **102** can be accomplished through, for example, a connector client **114** and connector server **116.** The agent side can also utilize a protocol adapter **118** to communicate with an application supporting JMX agents **120**. These tests can communicate directly with the MBean server **106** to access runtime MBeans. Runtime MBeans represent the runtime configuration and metrics of a domain as it is running. Runtime MBeans in this embodiment are not modifiable, but may be modified in other embodiments. In order to accurately capture values, a RuntimeMBean can be sampled at a frequency at least twice that of the signal being measured. Each test can then specify the desired testing frequency.

[0021] These tests can independently assess measurements made by the RuntimeMBean, in order to determine a pass/fail result. In a first technique, a carefully controlled test/server interaction can be used, such that the expected return value can be known beforehand. In a second technique a feature outside of the LS, such as the system time, can be used to measure a feature of the LS independent of its own internal mechanisms. This second technique is not exact, and therefore may include a margin of error. It will be evident to those skilled in the art that other similar or appropriate techniques may be used that are within the spirit and scope of the invention.

[0022] A class such as SignalGenerator may implement the SignalGeneratorMBean interface for all but one method, such as for example SignalGenerator::calculateValue(time : long). Subclasses of SignalGenerator, such as may implement the method CalculateValue, can create a time-varying signal that starts at time 0, when a method SignalGenerator::start is called. As an example, a sinusoidal signal may be implemented as sin(*wt*) where *t* is the time and *w* is the angular velocity.

[0023] A sequence diagram is shown in **Figure 2** for one such test **200** of the present invention. Figure 2 shows the dynamic runtime behavior of a class such as SquareWave **202**, StringGenerator **204**, or Ramp **206**. These classes may be subclasses that are derived from SignalGenerator **208**. **Figure 2** also shows a structural placement of the class SquareWave **202** within the test **200**. CounterMonitor **212** is a class that is responsible for polling the signal under observation. CounterMonitor **212** may, for instance, call a getCurrentObject or similar method of the SignalGenerator **208** class in order to get the current time. SignalGenerator **208** may then implement SignalGeneratorMBean **210**. A granularity period exists between successive polling of SquareWave **202** by CounterMonitor **212**. If one of the pollings exceeds a certain threshold value, CounterMonitor **212** may notify a notification listener.

[0024] GenericNotificationListener is one such test class that may be called by CounterMonitor when a signal being polled meets predefined conditions, such as the afore-mentioned threshold value, with the conditions being defined by CounterMonitor. GenericNotificationListener can also be responsible for storing values deemed important by CounterMonitor. TestCounterMonitor is an example of a class that can be called by an end user or another object or application in order to execute the test.

[0025] Another view of a testing process of the present invention is shown in **Figure 3**. In this test **300**, a tester **302** or end-user runs TestCounterMonitor **304**. During the setup portion **312**, TestCounterMonitor **304** initiates CounterMonitor **306**, which in turn polls SquareWave **310** and initiates GenericNotificationListener **308**. Once the setup portion **312** is complete, the testing is started **314** as CounterMonitor **306** polls SquareWave **310**, and continues polling **316** according to the granularity setting. In the event that a threshold or value is reached or exceeded, CounterMonitor **306** can notify **318** GeneritcNotificationListener **308** and can continue polling SquareWave **310**. After the polling process is complete, the process is stopped **320**. SquareWave **310** can then store the testing values, and TestCounterMonitor **304** can compare the results.

[0026] Table 1 shows some of the symbols that may be used in the signal equations, a description of each symbol, a preferred range of values for each symbol, and example values. Some of the signal equations using these symbols are shown in Table 2, as well as the subclasses that implement those signals. For example, if a signal is to be generated that increases monotonically over time, the Ramp signal equation can be used, which is equivalent to the basic equation for a line ($y = m * x + b$). In Ramp, m is the slope or rate of increase of the linear signal over time, while b is the y-intercept, or the initial value of the signal at time zero.

[0027] These equations may form a library of possible signals that can be generated, providing a significant level of flexibility to the system. A developer can produce new signals simply by selecting the proper equation and setting proper values for each variable in the selected equation. It may also be possible, if the library of equations is not sufficient, for the developer to enter a new equation into the library in order to generate a new signal type.

Table 1 - Preferred equation symbols

Symbol	Description	Type	Example
m	Multiplier, slope, or amplitude	scalar number, $-\infty < m < \infty$	4.87, 2, -78.9, 0.0
b	offset, y-intercept	scalar number, $-\infty < b < \infty$	4.87, 2, -78.9, 0.0
t	Time	scalar integer, $0 < t < \infty$	0, 14, 60
T	Period	scalar integer, $0 < T < \infty$	0, 14, 60
n	Iterations	scalar integer, $0 < n < \infty$	0, 1, 2, 3
f(t)	Function	Function of time	f(0) = 3, f(1) = 6, f(2) = 9
d(t)	unit impulse f'n	Function, $_{-\infty}\int^{\infty} \delta(t) = 1$	f(-1)= 0, f(0) $\equiv \infty$, f(1) = 0
$\sum$	Summation	Function	$_{n=0}{}^{3}\sum n = 6$
u(t)	unit step function	Function	u(-1)=0, u(0)=1, u(1)=1
s(t)	String array	Array	s(3)=StringArray[3]='hello'

Table 2 - example signals

Signal	**Parameters**	**Equation**
Ramp	b, m	f(t) = b + m * t
PeriodicRamp	b,m,T	f(t) = $\sum$(b+m * (t-nT)) u(t-nT)
AlternatingRamp	b,m,T	f(t) = (b + m*t) * $(-1)^{(2T+t)/T}$
SquareWave	b,m,T	f(t) = b+m[S(u(t-nT)-2u(t-(n+1/2)T)+u(t–(n+1)T)]
CarrierSquareWave	b,m,T,T'=1/10T	f(t) = b+m[$\sum$(u(t-nT)-2u(t-(n+1/2)T)+u(t–(n+1)T)+ $\sum$(1/10(u(t-nT')-2u(t-(n+1/2)T')+u(t–(n+1)T')]
StringGenerator	T, StringArray	f(t) = $\int\delta$(t - nT) s(t) = StringArray[f(t)]

[0028] A signal generator of the present invention may also include a generic user interface, or GUI. This can provide for an easy reconfiguration of the signals, as described above, by allowing a user or developer to simply select or enter an appropriate signal equation and variable values. There may be no need for the values used in the wave equations to be hard coded or stored in a source file, although at least some of the commonly used signals may be stored as such.

[0029] Another embodiment of the invention utilizes the signal generator as a string generator. This can be useful in applications where the accurate generation of a data stream, such as a stream of words, may be necessary for testing. In an example of a system in accordance with the present invention, a system may generate two strings, such as may represent "pass" and "fail." The application could determine the test signal that would be produced when transmitting that string of text, and the string generator could generate the appropriate signal. The monitor or notification listener could then store the signal as received, which could then be compared against the test signal to quickly determine the accuracy of the monitor.

[0030] Other features, aspects and objects of the invention can be obtained from a review of the figures and the claims. It is to be understood that other embodiments of the invention can be developed and fall within the spirit and scope of the invention and claims.

[0031] The foregoing description of preferred embodiments of the present invention has been provided for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise forms disclosed. Obviously, many modifications and variations will be apparent to the practitioner skilled in the art. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, thereby enabling others skilled in the art to understand the invention for various embodiments and with various modifications that are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the following claims and their equivalence.